Objektově orientované programování

Generičnost

2023/24

Osnova přednášky

- Generičnost – PROČ, KDY, JAK
- Příklad

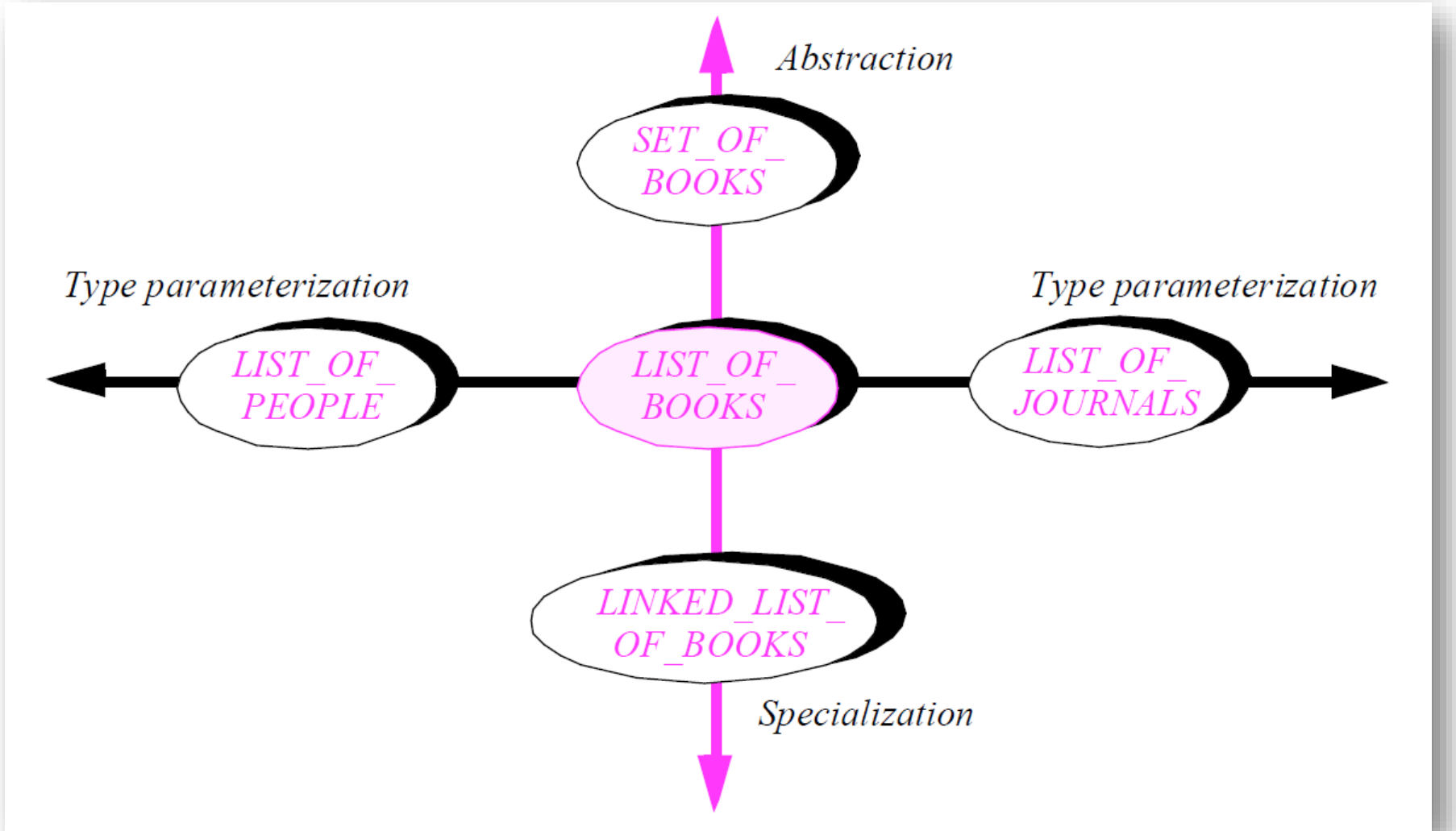
Generičnost – PROČ, KDY, JAK

Generičnost

- Generický – obecný, univerzální
  - opak – specifický
- Generičnost je možnost programovacího jazyka použít při definici na místě typů parametr.
  - Typy jsou použity podobně jako parametry.

Co se řeší?

- Opakovaná použitelnost, rozšiřitelnost, spolehlivost.
- Dědičnost poskytuje *abstrakci* danou *speciálními případy* tříd.
- Generičnost poskytuje *abstrakci* danou *parametrizací* tříd.


Abstraction
SET_OF_
BOOKS
Type parameterization
Type parameterization
LIST_OF_
PEOPLE
LIST_OF_
BOOKS
LIST_OF_
JOURNALS
LINKED_LIST_
OF_BOOKS
Specialization

Abstrakce #1

- *SET_OF_BOOKS* je nejobecnějším *kontejnerem* na knihy.
- *LIST_OF_BOOKS* je speciálním případem *SET_OF_BOOKS*
- *LINKED_LIST_OF_BOOKS* je speciálním případem *LIST_OF_BOOKS*.
- **Jde o abstrakci promítnutou do dědičnosti.**

Abstrakce #2

- *LIST_OF_BOOKS, LIST_OF_PEOPLE, LIST_OF_JOURNALS* jsou speciálními případy seznamu libovolných objektů.
- Všechny seznamy mají naprosto stejné chování (*put, get, remove,…*).
- **Opravdu?**
- Pracují s jinými parametry (*Book, Person, Journal*)!!!
- **Jde o abstrakci promítnutou do generičnosti.**

Dědičnost x Generičnost

- Společně řeší problém abstrakce. Jak jej řeší?
- Dědičnost *vertikálně* (hierarchie dědičnosti s rozšířením resp. změnou chování, jde o změnu – specializaci - typu)
- Generičnost *horizontálně* (zachovává funkčnost, pracuje se s různými typy parametrů).

Kde se generičnost hodí?

- Zejména pro implementaci abstraktních datových typů (zásobník, fronta, seznamy, stromy).
- Jde o struktury, které pro konkrétní použití předpokládají uložení objektů stejného typu.
- Musí ale fungovat pro libovolnou konkrétní třídu (typ).

Jak postupovat při návrhu?

1. Nejdříve deklarujeme, s jakou třídou se bude pracovat v generické třídě (generický parametr).

2. Poté napíšeme generickou třídu tak, že pracuje s tímto deklarovaným typem (typ je tedy parametrem).

3. Při použití dosadíme za deklarovaný typ konkrétní třídu.

Co lze s generickým parametrem dělat?

Uses of entities of a formal generic type

The valid uses for an entity *x* whose type *G* is a formal generic parameter are the following:

G1 • Use of *x* as left-hand side in an assignment, *x* := *y*, where the right-hand side expression *y* is also of type *G*.

G2 • Use of *x* as right-hand side of an assignment *y* := *x*, where the left-hand side entity *y* is also of type *G*.

G3 • Use of *x* in a boolean expression of the form *x* = *y* or *x* /= *y*, where *y* is also of type *G*.

G4 • Use of *x* as actual argument in a routine call corresponding to a formal argument declared of type *G*, or of type *ANY*.

G5 • Use as target of a call to a feature of *ANY*.

Příklad

Generická třída

- Třída *BOX* je generická …
- … a pracuje s generickým parametrem *T*
- Za *T* lze dosadit libovolnou třídu (typ)

```
template<class T>
class BOX {
private:
    T * instance;

public:
    BOX(T * i);
    T * GetInstance();
};

template<class T>
BOX<T>::BOX(T * i){
    this->instance = i;
}

template<class T>
T * BOX<T>::GetInstance(){
    return this->instance;
}
```

Co dosadíme za parametr?

- Třída (typ) A bude pro generickou třídu BOX parametrem.
- Dosazením typu se z generické třídy BOX stane třída specifická (pro typ A).

```
class A {
private:
    int value;

public:
    A(int v);
    int GetValue();
};

A::A(int v){
    this->value = v;
}

int A::GetValue(){
    return this->value;
}
```

Použití

```
int main() {
    A * a = new A(50);
    BOX<A> * ta = new BOX<A>(a);

    cout << ta->GetInstance()->GetValue();

    delete ta;
    delete a;

    getchar();
    return 0;
}
```


50

Přidejme další třídu (jen na zkoušku)

```
class B : public A{
public:
    B(int v);
};

B::B(int v) : A(v){
}
```

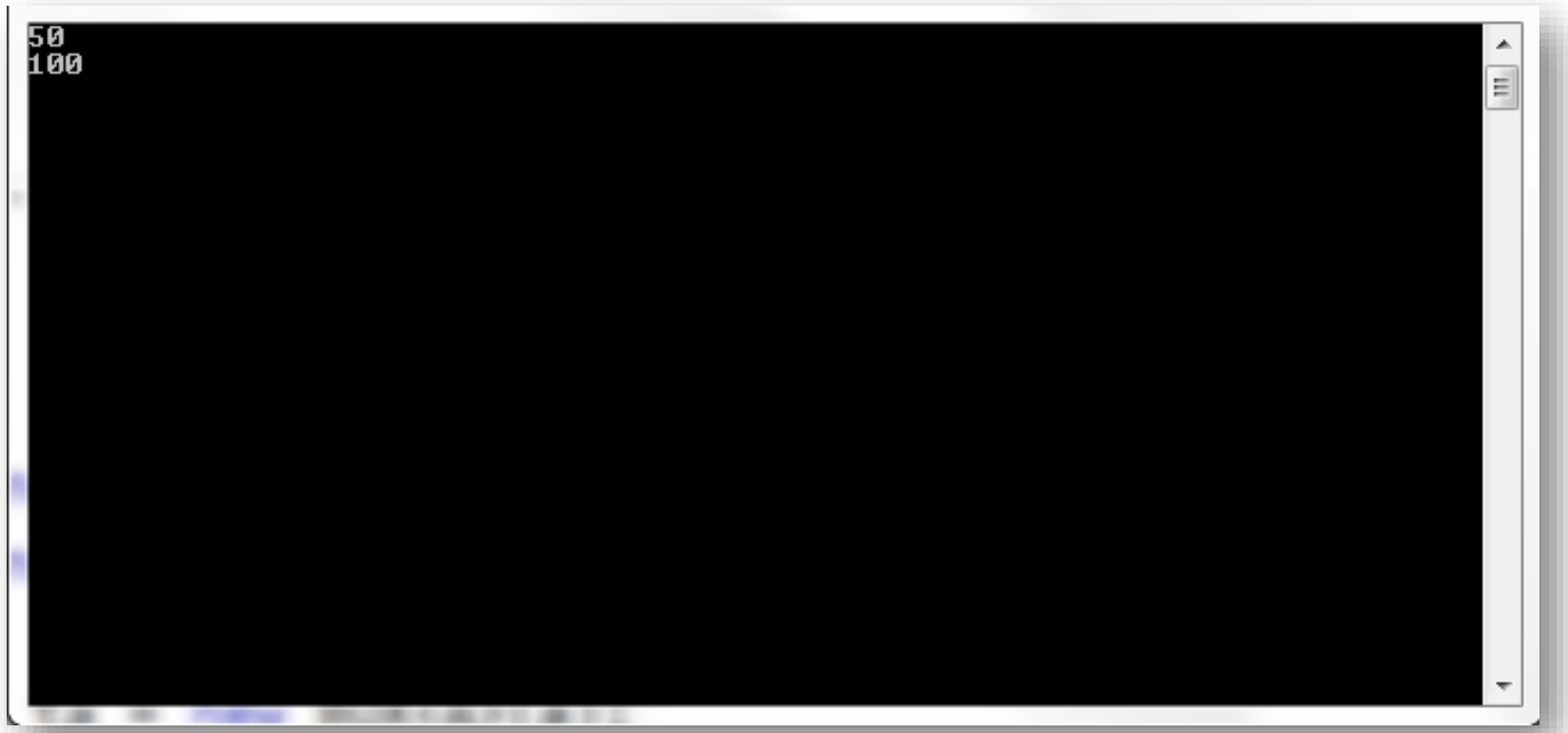
Co se vypíše?

```
A * a = new A(50);
B * b = new B(100);

BOX<A> * ta = new BOX<A>(a);
BOX<B> * tb = new BOX<B>(b);

cout << ta->GetInstance()->GetValue() << endl;
cout << tb->GetInstance()->GetValue() << endl;

delete ta;
delete tb;
delete a;
delete b;
```


50
100

A co teď? Jde to?

```
A * a = new A(50);
B * b = new B(100);

BOX<A> * ta = new BOX<A>(b);
BOX<B> * tb = new BOX<B>(b);

cout << ta->GetInstance()->GetValue() << endl;
cout << tb->GetInstance()->GetValue() << endl;

delete ta;
delete tb;
delete a;
delete b;
```


100
100

Omezení?

- Polymorfní datové struktury…
  - Stále platí, že pracujeme POUZE s chováním třídy, která je dosazena za generický parametr.
- Pokud využíváme v generické třídě chování generického parametru, pak dosazená třída musí toto chování mít!!!
  - Pokud pouze uchováváme a vracíme instance tříd dosazené za generický parametr, nic nehrozí.

Shrnutí

- Třídy mohou mít formální generické parametry, které reprezentují typ.
- Generické třídy slouží jako popis struktur, implementovaných stejným způsobem, a to bez ohledu na typ, se kterým pracují.
- Klient generické třídy musí poskytnout typ, se kterým se bude pracovat.
- Nesmí se pracovat s operacemi, které třída dosazená za generický parametr nemá.
- Generičnost má omezení dané polymorfismem!!!

Kontrolní otázky

- Co rozumíme pojmem generičnost a čím je reprezentována v C++?
- Vysvětlete rozdíl mezi abstrakcí založené na dědičnosti a generičnosti.
- Kdy je potřeba použít generičnost?
- Které operace je možno použít pro entity generického typu?
- Co je potřeba k návrhu generické třídy?
- Jaká jsou omezení generičnosti?
- Jak implementovat generickou třídu v C++ (deklarace i definice)?
- Kolik implementací generické třídy budeme mít, pokud použijeme deset jejích instancí, ale všechny jen pro jeden stejný generický parametr? Proč?

Ke studiu

- Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall 1997. [317-331]

Technické poznámky

- Generická třída může mít i více generických parametrů. Vymyslete příklad takové třídy a implementujte ji.

- I přesto, že máme jen jeden zdrojový kód pro každou generickou třídu, tak při jeho použití (překladu) pro různé generické parametry má každá generická třída se specifickým parametrem svou vlastní implementaci (překlad v paměti).

Úkoly na cvičení

- Implementujte příklad z prezentace a vyzkoušejte i kombinaci generičnosti s polymorfismem.
- Implementujte datové struktury jako je zásobník, fronta apod. jako generické typy.
- Napište příklad využívající vámi implementované generické typy.
- Popřemýšlejte, jak a kde využít generičnost v příkladech s bankou a s elektronickým obchodem a popř. doplňte implementaci o práci s generickými třídami.